Meta-Data Storage Conventions in MOAB

Timothy J. Tautges

INTRODUCTION

The Mesh-Oriented datABase (MOAB) is a library for representing finite element and other types of mesh data [1]. Various types of meta-data are often used in conjunction with a mesh. Examples include boundary condition groupings, material types, and provenance information for the mesh. Because the data model used in MOAB is so abstract, conventions are useful for describing how meta-data is stored into that data model. This document describes those conventions for several types of data commonly found in meshes stored in MOAB. Because the data models used by MOAB and iMesh, the ITAPS mesh interface [2], are so similar, the conventions described here apply almost unmodified to iMesh as well as to MOAB.

The meshes represented in MOAB originate in a variety of forms, including mesh read from files of various formats (e.g. CUBIT ".cub" file, VTK, etc.) as well as mesh written into MOAB directly by various software libraries (e.g. MeshKit). Although there is no standard for naming or storing meta-data with a mesh, there is a great deal of commonality in the types of meta-data typically found with mesh data. This document describes conventions that have been established for commonly encountered meta-data. Various mesh readers implemented in MOAB attempt to read meta-data from a file and write it into the MOAB data model using these conventions. Although there is no requirement to store a given type of meta-data in the form described here, a number of services have been written to handle meta-data using these conventions, no matter the source of the meta-data being processed.

Several specific tools are often used in concert with MOAB and bear special mention here. The CUBIT toolkit generates finite element meshes, and saves them to a native save file (referred to as a ".cub" file) which MOAB is able to read. Reading CUBIT meshes into MOAB through the .cub file format is preferred over other formats, since most other mesh formats written by CUBIT do not save most meta-data. The MeshKit library also generates mesh using CGM and MOAB, and uses the same conventions for storing meshes into MOAB. Finally, MOAB includes a CGM reader which can read a geometric model into a faceted representation in MOAB. Meta-data from all these tools are stored in MOAB using the conventions described here.

The MOAB data model consists of the following basic types:

- **Entity:** The basic elements of topology, e.g. vertex, edge, triangle, tetrahedron, etc. MOAB represents all types in the finite element zoo, plus polygons and polyhedra.
- **Entity Set:** An arbitrary collection of entities and other sets. Sets can have parent/child relations with other sets, and these relations are distinct from "contains" relations.
- **Interface:** The interface object through which other entities are accessed, in the sense of object-oriented-programming. iMesh refers to the interface as the "root" set.
- **Tag:** A piece of data that can be assigned a distinct value to each entity and entity set, and to the interface itself. Tags have a prescribed name, size in bytes, and data type; allowed data types are integer, double, entity handle, and byte or opaque.

The following section describes each meta-data tag convention in detail; these conventions are also

summarized in Table 1.

Meta-Data Conventions

Meta-data is stored in MOAB and iMesh in the form of tags applied to either entities or entity sets. For meta-data represented as entity sets, the contents of those sets are determined by the convention, with tags on those sets identifying them with the convention and adding any other semantic data.

Each meta-data convention is described in a subsection below. Each convention begins with a short description of:

- Whether tags associated with the convention are assigned to entities or entity sets
- The tag(s) associated with the convention; information for each tag includes the name, the data type (I=integer, D=double, C=character, H=handle), and the tag length. Tag lengths are specified after an asterisk (*); for example, C*32 implies a tag with character type and length 32. Unspecified lengths correspond to length one.

Name

(Data: Entity sets, entities; Tag(s): NAME/C*32)

Character strings are used in many different contexts in applications. MOAB uses the "NAME" tag to store character strings used to name entities. This tag is of byte-type and is of length 32 bytes. *Note that the string stored in this tag may or may not be terminated with a NULL character. It is always prudent account for missing NULL terminator, to avoid buffer overflow errors in the application.* Applications are free to define their own version of the NAME tag with a longer length, though this definition may conflict with other services attempting to use this tag with the conventional size. Applications needing a string tag with a longer or variable length can also use MOAB's variable-length tag type, though this will not be compatible with iMesh.

Global Identifier

(Data: Entity sets, entities; Tag(s): GLOBAL_ID/I)

Global identifiers are used in many different contexts in applications. Geometric model entities are identified by dimension and id, e.g. "Volume 1". Mesh vertices and elements are identified similarly in mesh generation codes. Boundary conditions and material types are identified similarly. This tag is used to store such information. This tag is currently stored in a 32-byte integer, though this may change in the future.

Geometric Model Information

(Data: Entity sets; Tag(s): GEOM_DIMENSION/I, GLOBAL_ID/I, NAME/C*32, CATEGORY/C*32)

Mesh generation is often performed starting from a geometric model, represented in some form of CAD engine. Many of the meshes used by MOAB are generated based on the CGM library. Geometric models contain both topological information (the topological entities in the geometric model) and shape information (the geometric shape of those entities), as well as other meta-data written to the entities in a model. When a mesh is read from a CUBIT .cub file, meta-data from the geometric model is read and represented in the MOAB data model, as described below. **Note that although MOAB reads and represents meta-data associated with the geometric model, it does not**

represent the geometric model itself. Therefore, shape-related information, e.g. the arc length of an edge or surface normal at a given point, can be retrieved only from the model represented in CGM or another geometric modeling engine.

The information contained in a geometric model, read into and represented in MOAB, consists of:

- Model entities (vertex, edge, face, volume)
- Topological relationships between model entities
- Groups of model entities
- Model entity/group ids
- Model entity/group names

The storage of this information into MOAB's data model is described for each type is described below.

Entities

Entities in the geometric model (VERTEX, EDGE, FACE, VOLUME) are each represented by an entity set[1]. These sets are tagged with the "GEOM_DIMENSION" tag, with integer value equal to the topological dimension of the entity (VERTEX = 0, EDGE = 1, etc.) These sets *contain* the mesh *owned* by the corresponding entity in the geometric model. Note this does not include mesh owned by bounding entities; thus, the set for a FACE will not contain the mesh vertices owned by bounding EDGEs in the geometric model. These sets may or may not contain mesh entities of intermediate dimension, e.g. mesh edges owned by a FACE or faces owned by a VOLUME, depending on the application generating the mesh or the file from which the mesh was read. These sets are all set-types, i.e. the order of entities in the sets is not significant, except in the case of EDGE sets, where order of the mesh vertices and edges corresponds to the relative order of vertices and edges at the time of mesh generation. In MOAB, these sets are non-tracking by default, i.e. entities do not have knowledge of which geometry sets they are members of.

Topological Relationships

In the geometric model, each FACE is bounded by zero or more EDGEs; other topological relationships between geometric entities exist in a similar manner. These relationships are embedded in the data model using parent/child relations between entity sets. For example, the entity set corresponding to a FACE will have child sets, each corresponding to a bounding EDGE, and parent sets, each corresponding to a VOLUME bounded by that FACE. The relative order of sets in those parent/child lists is not significant, thus, "loops" bounding a FACE cannot reliably be inferred from this data.

Groups

Geometric entities are sometimes assigned to application-specific groups. These groups are represented using entity sets, tagged with a "GROUP" tag whose value equals the group id. Group sets are "set"-type, and are not tracking sets. These sets *contain* the sets corresponding to geometric entities contained in the groups in the geometric model, as well as any mesh entities assigned to the group.

Material Type

(Data: Entity sets; Tag(s): MATERIAL_SET/I)

Most finite element and other PDE-based analysis codes require a material type for each cell or element in the simulation. MOAB uses entity sets to store this information, in the form of entity sets. The MATERIAL_SET tag is used to identify these sets. The value of this tag is conventionally an integer;

1 Body-type entities from CUBIT are not explicitly represented in MOAB.

in most cases this stores a user-assigned identifier associated with that material.

CUBIT assigns material types using what it calls “element blocks”, with each element block given a user-assigned id number and optionally a name. The CUBIT and Exodus file readers in MOAB read element blocks into MATERIAL_SET sets.

In CUBIT, materials are typically assigned by assigning geometric volumes to element blocks. Therefore, *material sets often contain entity sets corresponding to those volumes*. Thus, a materrial set in MOAB is unlikely to contain mesh entities directly; rather, that set contains other sets which contain mesh entities. In these cases, mesh entities can be retrieved by passing a “recursive” flag to the appropriate function (MOAB), or by calling the getEntitiesRec extension function (iMesh) provided by MOAB.

Boundary Conditions (Dirichlet, Neumann)

(Data: Entity sets; Tag(s): DIRICHLET_SET/I, NEUMANN_SET/I)

Boundary conditions are often specified in terms of geometric model entities, similar to material types. MOAB uses entity sets to store this information as well. The DIRICHLET_SET and NEUMANN_SET tags are used to represent Dirichlet- and Neumann-type boundary condition sets, resp. By convention, Neumann sets usually contain (indirectly) intermediate-dimension entities like edges in a 2D mesh or faces in a 3D mesh, while Dirichlet sets usually contain vertices. However, these conventions are not enforced in any way. Other than these characteristics, Dirichlet and Neumann sets are handled similarly to material sets.

Parallel Mesh Constructs

(Data: Entity sets, entities; Tag(s): PARALLEL_PART/I, PARALLEL_PARTITION/I, PSTATUS/C*1, PARALLEL_SHARED_PROC/I, PARALLEL/SHARED_HANDLE/H, PARALLEL_SHARED_PROCS/I*NP, PARALLEL_SHARED_HANDLES/H*NP)

On a parallel computer, MOAB can represent the mesh on each processor as well as information about entities shared with neighboring processors. Some of this information is also relevant even when the mesh is represented on a serial machine. MOAB uses several tag and set conventions to describe the parallel nature of a mesh. This information is summarized here; for a more complete description of MOAB’s parallel mesh representation and functionality, see [ref-moabpar].

Parallel partition, parts

Most parallel mesh applications use a domain decomposition approach, where each processor solves for a subset of the domain. The set of entities solved by a given processor is referred to as a part, and the collection of parts together is called the partition. MOAB stores each part in an entity set, marked with the PARALLEL_PART tag, whose value is the rank of the processor assigned that part; an entity set which contains all part sets is given the PARALLEL_PARTITION tag, whose value is currently meaningless. The MBZoltan tool included as a tool in MOAB can partition a mesh for parallel solution, and writes the partition to the mesh in the form of parts and partitions. Both these types of sets can be accessed in a serial mesh, e.g. for visualization.

Part interfaces

When a partitioned mesh has been loaded on a parallel computer, the part on a given processor may

share portions of its boundary with parts on other processors. These shared regions are called part interfaces, and are also represented using entity sets. These sets are marked with the PARALLEL_INTERFACE tag, whose value is currently meaningless.

Shared processor and handle

For entities shared between processors, it is helpful to know locally which other processor shares an entity, and what the entity's handle is on the remote processor. There are two cases which are useful to distinguish, first where an entity is shared with only one other processor (referred to as shared), and second when a processor is shared by more than one other processor (referred to as multi-shared). Shared entities are given the PARALLEL_SHARED_PROC and PARALLEL_SHARED_HANDLE tags, which store the rank of the sharing processor and the handle of the entity on that processor, respectively. Multi-shared entities are marked with the PARALLEL_SHARED_PROCS and PARALLEL_SHARED_HANDLES tags; these tags have a length NP assigned at compile time in MOAB, with default values of -1 for processor rank and zero for handle (which are each invalid values for the corresponding data). The processors/handles sharing a given entity are then written on the front of the arrays. So, for example, an entity on processor rank 0, shared by processors 1 and 2, would have a PARALLEL_SHARED_PROCS tag whose values would be [1, 2, -1, -1, …], with PARALLEL_SHARED_HANDLES values of [m, n, 0, 0, …], where m and n would be the handles of that entity on processors 1 and 2. The shared versions of these tags are "dense", with default values which denote unshared entities. The multi-shared tags are sparse tags in MOAB, with no default value.

Parallel status

In addition to the tags above, MOAB also defines the PSTATUS tag, whose bits contain information about the parallel status of a given entity. Starting with least significant bit, these bits represent whether an entity is 1) not owned, 2) shared, 3) multi-shared, 4) interface, 5) a ghost entity. The first bit being set indicates "not owned" so that the default value for this tag, of zero, corresponds to an owned, unshared entity, which will be the state of most entities on a given processor.

References

[1] T.J. Tautges, R. Meyers, K. Merkley, C. Stimpson, and C. Ernst, *MOAB: A Mesh-Oriented Database*, Sandia National Laboratories, 2004.

[2] L. Diachin, A. Bauer, B. Fix, J. Kraftcheck, K. Jansen, X. Luo, M. Miller, C. Ollivier-Gooch, M.S. Shephard, T. Tautges, and H. Trease, "Interoperable mesh and geometry tools for advanced petascale simulations," *Journal of Physics: Conference Series*, vol. 78, 2007, p. 012015.

Appendix A: Summary

Table 1: Summary of MOAB meta-data conventions.

Convention	Applies to (E=entity, S=set)	Tag(s) (type/length)	Description
Name	E, S	NAME/C*32	
Global identifier	E, S	GLOBAL_ID/I	
Geometric topology	S	GEOM_DIMENSION/I, GLOBAL_ID/I, NAME/C*32,	Sets contain mesh owned by that entity; parent/child links to

		CATEGORY/C*32	bounded/bounding entities in geometric model
Material type	S	MATERIAL_SET/I	Set contains entities or sets assigned a common material type
Boundary condition	S	DIRICHLET_SET/I NEUMANN_SET/I	Set contains entities or sets assigned a particular boundary condition; neumann sets usually contain edges (2D) or faces (3D)
Parallel mesh constructs	E, S	Various (see description)	Data which describes parallel mesh

*Table 2: Summary of MOAB conventional tag names, types, and purposes. Data types are I=integer, D=double, C=character, H=entity handle. Data type with *x denote length of x elements of that data type.*

Tag name	**Data type**	**Applies to (E=entity, S=set)**	**Purpose**
CATEGORY	C*32	S	String describing purpose of set; examples include “group”, “vertex”, “edge”, “surface”, “volume”
DIRICHLET_SET	I	S	Entities or sets with common boundary condition
GEOM_ DIMENSION	I	S	Identifies mesh entities resolving a given geometric model entity
GLOBAL_ID	I	E, S	Application-specific entity id
MATERIAL_SET	I	S	Entities or sets grouped by material type
NAME	C*32	E, S	User-assigned entity name(s); multiple names delimited with ?
NEUMANN_SET	I	S	Entities or sets with common boundary condition
PARALLEL_PART	I	S	Represent a part in a partition
PARALLEL_ PARTITION	I	S	Represents a partition of the mesh for parallel solution, which is a collection of parts
PARALLEL_SHARED_ PROC	I	E, S	Rank of other processor sharing this entity/set
PARALLEL_SHARED_ HANDLE	H	E, S	Handle of this entity/set on sharing processor
PARALLEL_SHARED_ PROCS	I*NP	E, S	Ranks of other processors sharing this entity/set

PARALLEL_SHARED_ HANDLES	H*NP	E, S	Handles of this entity/set on sharing processors
PSTATUS	C*1	E, S	Bit-field indicating various parallel information